Combinatorics, Probability and Computing (2019), **28**, pp. 777–790
doi:10.1017/S0963548319000075

CAMBRIDGE
UNIVERSITY PRESS

ARTICLE

A fast new algorithm for weak graph regularity

Jacob Fox[1,†], László Miklós Lovász[2*,‡] and Yufei Zhao[2§]

[1]Department of Mathematics, Stanford University, Stanford, CA 94305, USA and [2]Department of Mathematics, MIT, Cambridge, MA 02139, USA
*Corresponding author. Email: lmlovasz@mit.edu

(Received 12 January 2018; revised 18 November 2018; first published online 3 May 2019)

Abstract
We provide a deterministic algorithm that finds, in $\epsilon^{-O(1)}n^2$ time, an ϵ-regular Frieze–Kannan partition of a graph on n vertices. The algorithm outputs an approximation of a given graph as a weighted sum of $\epsilon^{-O(1)}$ many complete bipartite graphs.
As a corollary, we give a deterministic algorithm for estimating the number of copies of H in an n-vertex graph G up to an additive error of at most $\epsilon n^{v(H)}$, in time $\epsilon^{-O_H(1)}n^2$.

2010 MSC Codes: Primary 05C85; Secondary 05C50, 05D99

1. Introduction

The regularity method, based on Szemerédi's regularity lemma [18], is one of the most powerful tools in graph theory. Szemerédi [17] used an early version in the proof of his celebrated theorem on long arithmetic progressions in dense subsets of the integers. Roughly speaking, the regularity lemma says that every large graph can be partitioned into a small number of parts such that the bipartite subgraph between almost every pair of parts is random-like. One of the main drawbacks of the original regularity lemma is that it requires a tower-type number of parts, where the height of the tower depends on an error parameter ϵ. However, for many applications, the full power of the regularity lemma is not needed, and a weaker notion of Frieze–Kannan regularity suffices.

To state the regularity lemmas requires some terminology. Let G be a graph, and let X and Y be (not necessarily disjoint) vertex subsets. Let $e(X, Y)$ denote the number of pairs of vertices $(x, y) \in X \times Y$ that are edges of G. The *edge density* $d(X, Y) = e(X, Y)/(|X||Y|)$ between X and Y is the fraction of pairs in $X \times Y$ that are edges. The pair (X, Y) is *ϵ-regular* if, for all $X' \subseteq X$ and $Y' \subseteq Y$ with $|X'| \geqslant \epsilon|X|$ and $|Y'| \geqslant \epsilon|Y|$, we have $|d(X', Y') - d(X, Y)| < \epsilon$. Qualitatively, a pair of parts is ϵ-regular with small ϵ if the edge densities between pairs of large subsets are all roughly the same. A vertex partition $V = V_1 \cup \ldots \cup V_k$ is *equitable* if the parts have size as equal as possible, that is, we have $||V_i| - |V_j|| \leqslant 1$ for all i, j. An equitable vertex partition with k parts is *ϵ-regular* if all but ϵk^2 pairs of parts (V_i, V_j) are ϵ-regular. The regularity lemma states that for every $\epsilon > 0$ there is a (least) integer $K(\epsilon)$ such that every graph has an ϵ-regular equitable vertex partition into at most $K(\epsilon)$ parts.

[†]J. Fox is supported by a Packard Fellowship, by NSF CAREER award DMS 1352121, and by an Alfred P. Sloan Fellowship.
[‡]L. M. Lovász is supported by NSF Postdoctoral Fellowship Award DMS 1705204.
[§]Y. Zhao is supported by NSF awards DMS-1362326 and DMS-1764176, and the MIT Solomon Buchsbaum Fund.

To state Frieze–Kannan regularity precisely, we first extend the definition of $e(X, Y)$ and $d(X, Y)$ to weighted graphs. Below, by *weighted graph* we mean a graph with edge-weights. Given two sets of vertices X and Y, we let $e(X, Y)$ denote the sum of the edge-weights over pairs $(x, y) \in X \times Y$ (taking 0 if a pair does not have an edge). Let $d(X, Y) = e(X, Y)/(|X||Y|)$ as earlier. Recall that the cut metric $d_\square$ between two graphs G and H on the same vertex set $V = V(G) = V(H)$ is defined by

$$d_\square(G, H) := \max_{U,W \subseteq V} \frac{|e_G(U, W) - e_H(U, W)|}{|V|^2},$$

and this extends to graphs with weighted edges, and can be adapted to bipartite graphs (with given bipartitions). Given any edge-weighted graph G and any partition $\mathcal{P}\colon V = V_1 \cup V_2 \cup \cdots \cup V_t$ of the vertex set of G into t parts, let $G_\mathcal{P}$ denote the weighted graph with vertex set V obtained by giving weight $d_{ij} := d(V_i, V_j)$ to all pairs of vertices in $V_i \times V_j$, for every $1 \leqslant i \leqslant j \leqslant t$. We say $\mathcal{P}$ is an *ϵ-regular Frieze–Kannan* (or *ϵ-FK-regular*) partition if $d_\square(G, G_\mathcal{P}) \leqslant \epsilon$. In other words, $\mathcal{P}$ is an ϵ-regular Frieze–Kannan partition if

$$\left| e(S, T) - \sum_{i,j=1}^{t} d_{ij}|S \cap V_i||T \cap V_j| \right| \leqslant \epsilon |V|^2, \tag{1.1}$$

for all $S, T \subseteq V$. We say that sets S and T *witness* that $\mathcal{P}$ is not ϵ-FK-regular if the above inequality is violated.

Frieze and Kannan [7, 8] proved the following regularity lemma.

Theorem 1.1. (Frieze–Kannan). *Let $\epsilon > 0$. Every graph has an ϵ-regular Frieze–Kannan partition with at most $2^{2/\epsilon^2}$ parts.*

There is a variant of the weak regularity lemma, where the final output is not a partition of V into $2^{\epsilon^{-O(1)}}$ parts, but rather an approximation of the graphs as a sum of $\epsilon^{-O(1)}$ complete bipartite graphs, each assigned some (not necessarily non-negative) weight: see [8]. For $S, T \subseteq V$, we let $K_{S,T}$ denote the weighted graph where an edge $\{s, t\}$ has weight 1 if $s \in S$ and $t \in T$ (and weight 2 if $s, t \in S \cap T$) and weight zero otherwise. For any $c \in \mathbb{R}$, by cG we mean the weighted graph obtained from G by multiplying every edge-weight by c. For a pair of weighted graphs G_1, G_2 on the same set of vertices, we will use the notation $G_1 + G_2$ to denote the graph on the same vertex set with edge weights summed (and weight 0 corresponding to not having an edge). Additionally, we write c to mean the constant graph with all edge-weights equal to c. We also use $d(G) := d(V(G), V(G))$ to denote the edge density of the weighted graph G.

Theorem 1.2. (Frieze–Kannan). *Let $\epsilon > 0$. Let G be any weighted graph with $[-1, 1]$-valued edge weights. There exists an $r = O(\epsilon^{-2})$, and there exist subsets $S_1, \ldots, S_r, T_1, \ldots, T_r \subseteq V$, and $c_1, \ldots, c_k \in [-1, 1]$, so that*

$$d_\square(G, c_1 K_{S_1,T_1} + \cdots + c_r K_{S_r,T_r}) \leqslant \epsilon.$$

See [11, Lemma 4.1] for a simple proof (given there in a more general setting of arbitrary Hilbert spaces). It is well known using the triangle inequality (see *e.g.* [8]) that given sets and numbers as in the theorem, the common refinement of all S_i, T_i must be a 2ϵ-regular Frieze–Kannan partition.

In addition to proving that a partition or 'cut graph decomposition' exists, Frieze and Kannan gave probabilistic algorithms for finding a weak regular partition [7, 8] or decomposition. Two deterministic algorithms were given by Dellamonica, Kalyanasundaram, Martin, Rödl and Shapira [2, 3]. Specifically, in [2] they gave an $\epsilon^{-6}n^{\omega+o(1)}$ time algorithm ($\omega < 2.373$ is the matrix multiplication exponent) to generate an equitable ϵ-regular Frieze–Kannan partition of a graph on n

vertices into at most $2^{O(\epsilon^{-7})}$ parts. In [3] they gave a different algorithm, which improved the dependence of the running time on n from $O_\epsilon(n^{\omega+o(1)})$ to $O_\epsilon(n^2)$ while sacrificing the dependence of ϵ. Namely, it was shown that there is a deterministic algorithm that finds, in $2^{2^{\epsilon^{-O(1)}}} n^2$ time, an ϵ-regular Frieze–Kannan partition into at most $2^{\epsilon^{-O(1)}}$ parts.

In Section 2 we give an optimal algorithm that provides the best of both worlds: we give an algorithm that finds, in $\epsilon^{-O(1)} n^2$ time, a weakly regular partition.[1] In fact, we provide an algorithm for finding a cut graph decomposition, which is more useful in some applications. The algorithm is also self-contained.

Theorem 1.3. *There is a deterministic algorithm that, given $\epsilon > 0$ and an n-vertex graph G, outputs, in $\epsilon^{-O(1)} n^2$ time, subsets $S_1, S_2, \ldots, S_r, T_1, T_2, \ldots, T_r \subseteq V(G)$ and*

$$c_1, c_2, \ldots, c_r \in \left\{ -\frac{\epsilon^8}{300}, \frac{\epsilon^8}{300} \right\}$$

for some $r = O(\epsilon^{-16})$, such that

$$d_\square(G, c_1 K_{S_1,T_1} + \cdots + c_r K_{S_r,T_r}) \leqslant \epsilon.$$

Remark. The same proof also gives a bipartite version of the theorem. Namely, there is a deterministic algorithm that, given $\epsilon > 0$ and a graph G between two sets of vertices V and U of size n, outputs, in $\epsilon^{-O(1)} n^2$ time, subsets $S_1, S_2, \ldots, S_r \subseteq V$, $T_1, T_2, \ldots, T_r \subseteq U$ and

$$c_1, c_2, \ldots, c_r \in \left\{ -\frac{\epsilon^8}{300}, \frac{\epsilon^8}{300} \right\}$$

for some $r = O(\epsilon^{-16})$, such that

$$d_\square(G, c_1 K_{S_1,T_1} + \cdots + c_r K_{S_r,T_r}) \leqslant \epsilon.$$

Remark. Given a decomposition as above, we can obtain a 2ϵ-regular Frieze–Kannan partition in $O(nr)$ time via the common refinement of all S_i, T_i. Indeed, the parts of the partitions are represented by $2r$-bit strings indicating membership in the sets $S_1, \ldots, S_r, T_1, \ldots, T_r$. We can sort the vertices with respect to their $2r$-bit strings lexicographically, by going through these $2r$ bits and sorting the vertices one bit at a time, each bit taking $O(n)$ time. This would sort the vertices into intervals corresponding to common refinement partitions of the sets S_i, T_i (a more naive implementation of taking refinements might take $O(n2^r)$ time, which is more than necessary).

In Section 3, using the above algorithmic weak regularity lemma, we obtain a deterministic algorithm for approximating the number of copies of a fixed vertex graph H in a large vertex graph G.[2] Note that there is an easy randomized algorithm for estimating the number of copies of H by sampling. However, it appears to be non-trivial to estimate this quantity deterministically. Duke, Lefmann and Rödl [4] gave an approximation algorithm for the number of copies of a k-vertex graph H in an n-vertex graph G up to an error of at most ϵn^k in time $O(2^{(k/\epsilon)^{O(1)}} n^{\omega+o(1)})$. We give a new algorithm which significantly improves the running time dependence on both n and ϵ.

Theorem 1.4. *There is a deterministic algorithm that, given $\epsilon > 0$, a graph H, and an n-vertex graph G, outputs, in $O(\epsilon^{-O_H(1)} n^2)$ time, the number of copies of H in G up to an additive error of at most $\epsilon n^{v(H)}$.*

[1]Theorem 1.3 replaces [6, Corollary 3.5], which we retracted [5].
[2]Theorem 1.4 replaces [6, Theorem 1.4], which we retracted [5].

Remark. An examination of the proof shows that the exponent of ϵ^{-1} in the running time can be $9^{|H|}$ (though it is not believed to be optimal). For example, we can count the number of cliques of order 1000 in an n-vertex graph up to an additive error $n^{1000-10^{-1000000}}$ in time $O(n^{2.1})$.

Remark. All results here can be generalized easily to weighted graphs G with bounded edge-weights.

2. Algorithmic weak regularity

Here we prove Theorem 1.3. We will prove the following roughly equivalent form. In order to state it, we first give some notation. Given a matrix A, we let $\|A\|$ denote the spectral norm, *i.e.* the largest singular value. It is well known that this is equal to the operator norm of A when viewed as an operator between L^2-spaces. We also use the Frobenius norm,

$$\|A\|_F = \sqrt{\sum_{i,j} a_{i,j}^2},$$

and the entry-wise maximum norm,

$$\|A\|_{\max} = \sup_{i,j} |a_{i,j}|.$$

Given a set $S \subseteq [n]$, we will let $\mathbf{1}_S \in \mathbb{R}^n$ denote the characteristic vector of S.

Theorem 2.1. *There is an algorithm that, given an $\epsilon > 0$ and a matrix $A \in [-1, 1]^{n\times n}$, outputs, in $\epsilon^{-O(1)}n^2$ time, subsets $S_1, \ldots, S_r, T_1, \ldots, T_r \subseteq [n]$ and real numbers*

$$c_1, \ldots, c_r \in \left\{-\frac{\epsilon^8}{300}, \frac{\epsilon^8}{300}\right\}$$

for some $r = O(\epsilon^{-16})$, such that, setting

$$A' = \sum_{i=1}^{r} c_i \mathbf{1}_{S_i} \mathbf{1}_{T_i}^\top,$$

each row and column of $A - A'$ has L^2-norm at most $\sqrt{n}$ (i.e. the sum of the squares of the entries is at most n), and

$$\|A - A'\| \leqslant \epsilon n.$$

If G and H are weighted graphs on a set X of size n, and A_G, A_H are the adjacency matrices, then

$$d_\square(G, H) \leqslant \frac{\|A_G - A_H\|}{n}.$$

Indeed, for any $S, T \subseteq [n]$, taking the characteristic vectors $\mathbf{1}_S$ and $\mathbf{1}_T$, we have

$$|e_G(S, T) - e_H(S, T)| = |\mathbf{1}_S^\top (A_G - A_H)\mathbf{1}_T| \leqslant \|A_G - A_H\| \|\mathbf{1}_S\|_2 \|\mathbf{1}_T\|_2 \leqslant \|A_G - A_H\| n.$$

Therefore, this theorem indeed implies Theorem 1.3 (taking A to be A_G). For the bipartite version (see the remark following Theorem 1.3), we can take A to be the (not necessarily symmetric) adjacency matrix where rows correspond to one vertex part and columns to the other vertex part.

The proof of the Frieze–Kannan regularity lemma and its algorithmic versions, roughly speaking, run as follows.

- Given a partition (starting with the trivial partition with one part), either it is ϵ-FK-regular (in which case we are done) or we can exhibit some pair of subsets S, T of vertices that witness the irregularity (in the algorithmic versions, one may only be guaranteed to find S and T that witness irregularity for some smaller value of ϵ).
- Refine the partition by using S and T to split each part into at most four parts, thereby increasing the total number of parts by a factor of at most 4.
- Repeat. Use a mean square density increment argument to upper-bound the number of possible iterations.

This can be modified to prove the approximation version. Roughly speaking, to find the appropriate S_i, T_i, c_i, in the second step of the above outline of the proof of the weak regularity lemma, instead of using S and T to refine the existing partition, we subtract $c\mathbf{1}_S\mathbf{1}_T^\top$ from the remaining matrix, for a carefully chosen c. We record the corresponding S_i, T_i, c_i in step i of this iteration. We can bound the number of iterations by observing that the L^2-norm of $A - c_1\mathbf{1}_{S_1}\mathbf{1}_{T_1}^\top - \cdots - c_i\mathbf{1}_{S_i}\mathbf{1}_{T_i}^\top$ must decrease by a certain amount at each step.

As for the algorithmic versions, the main challenge is checking whether a partition is regular, or a cut graph approximation is close in cut distance. Given a matrix A, up to a polynomial change in ϵ, having small singular values as a fraction of n is equivalent to $\operatorname{tr} AA^\top AA^\top$ being small as a fraction of n^4, which roughly says that most scalar products of rows are small as a fraction of n. Alon *et al.* [1] use this fact to obtain an algorithm which runs in $O(n^{\omega+o(1)})$ time, and either correctly states that a pair of parts is ϵ-regular or gives a pair of subsets which realizes it is not $\epsilon^{O(1)}$-regular. Dellamonica *et al.* [2] adapted this to the weak regular setting. Kohayakawa *et al.* [10] noticed that it suffices to check the scalar products along the edges of a well-chosen expander, which has a linear number of edges in terms of n, allowing them to obtain an $O_\epsilon(n^2)$-time algorithm. This was also the main idea of Dellamonica *et al.* [3], but their algorithm has running time doubly exponential in ϵ^{-1}. A further challenge in proving Theorem 2.1 with the cut matrix approximation is that the entries of the approximation matrices may not stay bounded, which was used in the algorithms for checking regularity. This is problematic, because for a general matrix A, the singular value (divided by n) and the cut-norm may be quite different. To counter this, we give an algorithm which checks regularity effectively under a weaker assumption that simply the L^2-norm of each row and each column stays bounded. Heuristically, the reason this property is useful is that it implies that if we have a singular vector (with norm 1) with a relatively large singular value, then no entry can be 'too large' – it must be 'spread out', which can then be used to show that a large singular value implies a large cut-norm. We then show that if we are careful, we can make sure that this property holds throughout the process.

Let us state this more precisely. Given a matrix A, let $\mathbf{a}_i$ be the ith row of A and $\mathbf{a}^j$ the jth column. Our main ingredient then is the following theorem. Note that in the algorithm below, the parameter C affects the running time but not the discrepancy of the output sets S, T.

Theorem 2.2. *There exists a $(C/\epsilon)^{O(1)}n^2$ algorithm which, given a matrix $A \in \mathbb{R}^{n\times n}$ such that $\|A\|_{\max} \leqslant C$, and each $\|\mathbf{a}_i\|_2^2 \leqslant n$, $\|\mathbf{a}^j\|_2^2 \leqslant n$ (or equivalently $\|A^\top A\|_{\max}, \|AA^\top\|_{\max} \leqslant n$), either*

- *correctly outputs that each singular value of A is at most ϵn, or*
- *outputs sets $S, T \subseteq [n]$ such that*

$$\left|\sum_{i\in S,k\in T} a_{i,k}\right| \geqslant \frac{\epsilon^8}{100}n^2.$$

(This implies that A has a singular value that is at least $(\epsilon^8/100)n$.)

In the next lemma, we construct the expander along which we will check the scalar products. For an integer n, let J_n denote the $n \times n$ matrix with each entry equal to 1.

Lemma 2.3. *There exist fixed absolute constants $l > 0$ and $0 < c < 1$ such that there is an algorithm which, given d_0 and n, outputs a matrix M on $\mathbb{R}^{n\times n}$ with non-negative integer entries, and an integer d with $d_0 \leqslant d \leqslant ld_0$, such that*

$$\left\| \frac{d}{n} J_n - M \right\| \leqslant d^{1-c}.$$

In other words, for any vector $\mathbf{v} = (v_i)_{i=1}^n \in \mathbb{R}^n$, we have

$$\left| \left(\sum_i v_i \right)^2 - \frac{n}{d} \mathbf{v}^\top M \mathbf{v} \right| \leqslant \frac{n}{d^c} \|\mathbf{v}\|_2^2. \tag{2.1}$$

The running time of the algorithm is $O(dn(\log n)^{O(1)})$.

Proof. Construct an l-regular two-sided expander G_0 on $[\widetilde{n}]$ for some $n \leqslant \widetilde{n} \leqslant Kn$ with K fixed. This can be done in $n(\log n)^{O(1)}$ time. For example, Margulis [13] constructed an 8-regular expander on $\mathbb{Z}_m \times \mathbb{Z}_m$ for every m, and Gabber and Galil [9] showed that all other eigenvalues (besides 8 with multiplicity 1) are at most $5\sqrt{2} < 8$. For every vertex $(x, y) \in \mathbb{Z}_m \times \mathbb{Z}_m$, its eight neighbours are

$$(x \pm 2y, y), (x \pm (2y + 1), y), (x, y \pm 2x), (x, y \pm (2x + 1)).$$

Therefore we can compute, for each vertex, a list of neighbours in time $O(\log m) = O(\log n)$, which then takes $O(n \log n)$ time total. Alternatively, we can start with a Ramanujan graph for some fixed degree, constructed explicitly by Lubotzky, Phillips and Sarnak [12], Margulis [14] and Morgenstern [15].

The adjacency matrix A_{G_0} has $A_{G_0}\mathbf{1} = l\mathbf{1}$, and all eigenvalues besides l have absolute value at most some explicit $a < l$. Let k be the integer and $\widetilde{M} = A_{G_0}^k$ be such that $d_0 n/\widetilde{n} \leqslant \widetilde{d} := l^k < ld_0 n/\widetilde{n}$. Note that $\widetilde{M}$ is symmetric and has non-negative integer entries, so it is the adjacency matrix of some graph G (possibly with multiple edges and loops). Clearly $\widetilde{M}\mathbf{1} = \widetilde{d}\mathbf{1}$, so $\widetilde{d}$ is an eigenvalue of $\widetilde{M}$, and all other eigenvalues have absolute value at most $a^k = a^{\log_l(\widetilde{d})} = \widetilde{d}^{\log_l(a)}$. Since $a < l$, $\widetilde{c} := 1 - \log_l(a) > 0$. This implies that

$$\left\| \frac{\widetilde{d}}{\widetilde{n}} J_{\widetilde{n}} - \widetilde{M} \right\| \leqslant \widetilde{d}^{1-\widetilde{c}}.$$

Take any set of n vertices, let M be the restricted submatrix of $\widetilde{M}$, and let $d = \widetilde{d}n/\widetilde{n}$. As $\widetilde{n}/n \leqslant K$, and the spectral norm of a matrix cannot increase when taking a submatrix, we have that

$$\left\| \frac{d}{n} J_n - M \right\| \leqslant \left\| \frac{\widetilde{d}}{\widetilde{n}} J_{\widetilde{n}} - \widetilde{M} \right\| \leqslant \widetilde{d}^{1-\widetilde{c}} = \left(\frac{\widetilde{n}}{n} d \right)^{1-\widetilde{c}} \leqslant (Kd)^{1-\widetilde{c}} \leqslant d^{1-c}$$

for an explicit $c > 0$.

We can construct G_0 in time $(\log n)^{O(1)} n$. We make sure, for each vertex, to keep a list of its neighbours. We then compute $A_{G_0}^i$ for $i = 1, 2, \ldots, k$. In each case, we make sure to keep a list of the l^i neighbours of each vertex (with multiplicities). We can then compute $A_{G_0}^{i+1}$ in $O(l^i n)$ time by computing the list of l^{i+1} neighbours for each vertex, by looking at its l neighbours in G_0 and taking the (multiset) union. The total running time is therefore

$$O\left((\log n)^{O(1)} n + \sum_{i=1}^{k} l^i n \right) = O(((\log n)^{O(1)} + d)n). \qquad \square$$

Alternatively, we could have used the zig-zag construction of expanders due to Reingold, Vadhan and Wigderson [16].

Proof of Theorem 2.2. Throughout this proof, we use the convention that i and j refer to rows and k and l refer to columns. The basic idea of the algorithm is as follows. It is easy to see that

$$\operatorname{tr}(AA^\top AA^\top) = \sum_{i,j,k,l} a_{i,k}a_{i,l}a_{j,k}a_{j,l}. \tag{2.2}$$

In order to estimate this sum, we can use the expander to only compute the sum for pairs (i, j) which form an edge of the expander (and then multiply by n/d). In fact, this is true even for the terms in (2.2) corresponding to a fixed k, l. We can therefore use the expander to estimate the sum in (2.2), and if it is large, find a k for which the sum of the terms corresponding to k are large. This will allow us to find sets S, T as required.

Here is the algorithm.

1. Construct the matrix M via Lemma 2.3 that satisfies (2.1) (inputting $d_0 = (3C^2\epsilon^{-4})^{1/c}$). Let $M = (m_{i,j})_{i,j=1}^n$.
2. For each i, j with $m_{i,j} > 0$, compute $s_{i,j} = \langle \mathbf{a}_i, \mathbf{a}_j \rangle$.
3. For each $k \in [n]$, compute

$$b_k = \sum_{i,j=1}^n m_{i,j}a_{i,k}a_{j,k}s_{i,j}.$$

4. If each $b_k \leqslant \frac{2}{3}\epsilon^4 dn^2$, return that $\|A\| \leqslant \epsilon n$.
5. If some $b_k \geqslant \frac{2}{3}\epsilon^4 dn^2$, do the following.
   a. Compute for each l

$$c_l = \sum_i a_{i,k}a_{i,l}.$$

   b. Let T be either the set of l such that $c_l > 0$, or the set of l such that $c_l < 0$, whichever has a bigger sum in absolute value.
   c. Compute for each $i \in [n]$ the values

$$d_T(i) = \sum_{k\in T} a_{i,k}.$$

   d. Let S be either the set of $i \in [n]$ such that $d_T(i) > 0$ or the set of $i \in [n]$ such that $d_T(i) < 0$, whichever has a bigger sum in absolute value.

Let us first analyse the running time. We can construct M in time $(\log n)^{O(1)}dn$. We can compute each $s_{i,j}$ in $O(n)$ time, so computing all of them takes $O(dn^2)$ time in total. Computing each b_k then similarly takes $O(dn)$ time (since we only need to sum the terms where $m_{i,j} > 0$, and we keep a list of these entries), so that takes $O(dn^2)$ total time. If the algorithm says that $\|A\| \leqslant \epsilon n$, then we are done. Otherwise, computing each c_l can be done in time $O(n)$, so that takes $O(n^2)$ time in total. We then obtain T in $O(n)$ time. Computing S then similarly takes $O(n^2)$ time. Since $d = (C/\epsilon)^{O(1)}$, this shows that the algorithm runs in time $(C/\epsilon)^{O(1)}n^2$.

We now show that the algorithm is correct. First, we show the following lemma, which makes precise that we can use the expander to estimate the sum (2.2).

Lemma 2.4. *For any $k, l \in [n]$, we have*

$$\left|\sum_{i,j} a_{i,k}a_{i,l}a_{j,k}a_{j,l} - \frac{n}{d}\sum_{i,j} m_{i,j}a_{i,k}a_{i,l}a_{j,k}a_{j,l}\right| \leqslant \frac{C^2n^2}{d^c} \leqslant \frac{\epsilon^4}{3}n^2. \tag{2.3}$$

Proof. Let $\mathbf{a}_{k,l}$ be the vector with entries $(\mathbf{a}_{k,l})_i = a_{i,k}a_{i,l}$. Since each $|a_{i,j}| \leqslant C$, we have that $\|\mathbf{a}_{k,l}\|_2^2 \leqslant C^2 n$. Therefore, by (2.1),

$$\left|\left(\sum_i a_{i,k}a_{i,l}\right)^2 - \frac{n}{d}\mathbf{a}_{k,l}^\top M\mathbf{a}_{k,l}\right| \leqslant \frac{C^2n^2}{d^c}.$$

Clearly

$$\left(\sum_i a_{i,k}a_{i,l}\right)^2 = \sum_{i,j} a_{i,k}a_{i,l}a_{j,k}a_{j,l},$$

and by the definition of M and $\mathbf{a}_{k,l}$, we have

$$\mathbf{a}_{k,l}^\top M\mathbf{a}_{k,l} = \sum_{i,j} m_{i,j}a_{i,k}a_{i,l}a_{j,k}a_{j,l}.$$ □

Lemma 2.5. *If the algorithm returns that $\|A\|_2 \leqslant \epsilon n$, then it is correct.*

Proof. We have

$$\sum_{k,l}\sum_{i,j} m_{i,j}a_{i,k}a_{i,l}a_{j,k}a_{j,l} = \sum_{i,j} m_{i,j}\langle \mathbf{a}_i, \mathbf{a}_j\rangle^2 = \sum_k\sum_{i,j} m_{i,j}a_{i,k}a_{j,k}\langle \mathbf{a}_i, \mathbf{a}_j\rangle = \sum_k b_k \leqslant \frac{2}{3}\epsilon^4 dn^3.$$

Summing (2.3) over all pairs $k, l \in [n]$, we have

$$\left|\sum_{k,l}\sum_{i,j} a_{i,k}a_{i,l}a_{j,k}a_{j,l} - \frac{n}{d}\sum_{k,l}\sum_{i,j} m_{i,j}a_{i,k}a_{i,l}a_{j,k}a_{j,l}\right| \leqslant \frac{\epsilon^4}{3}n^4.$$

Therefore,

$$\operatorname{tr} AA^\top AA^\top = \sum_{i,j,k,l} a_{i,k}a_{i,l}a_{j,k}a_{j,l} \leqslant \frac{n}{d}\sum_{i,j} m_{i,j}\langle \mathbf{a}_i, \mathbf{a}_j\rangle^2 + \frac{\epsilon^4}{3}n^4 \leqslant \epsilon^4 n^4.$$

Since $\operatorname{tr} AA^\top AA^\top$ is the sum of the fourth powers of the singular values, this implies that each singular value is at most ϵn. □

Lemma 2.6. *If the algorithm returns S and T, then*

$$\left|\sum_{(i,l)\in S\times T} a_{i,l}\right| \geqslant \frac{\epsilon^8}{100}n^2.$$

Proof. First, note that for the particular k we obtain in the algorithm, we have

$$\frac{2}{3}\epsilon^4 dn^2 \leqslant b_k = \sum_{i,j} m_{i,j}a_{i,k}a_{j,k}s_{i,j} = \sum_{i,j,l} m_{i,j}a_{i,k}a_{j,k}a_{i,l}a_{j,l}.$$

We claim that we have

$$\sum_{i,j,l\in[n]} a_{i,k}a_{j,k}a_{i,l}a_{j,l} \geqslant \frac{n}{d}\sum_{i,j,l\in[n]} m_{i,j}a_{i,k}a_{j,k}a_{i,l}a_{j,l} - \frac{\epsilon^4}{3}n^3 \geqslant \frac{\epsilon^4}{3}n^3.$$

Indeed, for any fixed l, by (2.3), we have

$$\left|\sum_{i,j\in[n]} a_{i,k}a_{j,k}a_{i,l}a_{j,l} - \frac{n}{d}\sum_{i,j\in[n]} m_{i,j}a_{i,k}a_{j,k}a_{i,l}a_{j,l}\right| \leqslant \frac{\epsilon^4}{3}n^2,$$

and we can add this up over all $l \in [n]$. Let $\mathbf{u} = (a_{i,k})_{i=1}^n$, and let $\mathbf{v}$ be the vector with coordinates

$$v_l = \sum_j a_{j,k}a_{j,l}.$$

Then $\|\mathbf{v}\|_\infty \leqslant n$ and $\|\mathbf{u}\|_2 \leqslant \sqrt{n}$, and we have

$$\mathbf{u}^\top A\mathbf{v} \geqslant \frac{\epsilon^4}{3}n^3.$$

Note, however, that

$$|\mathbf{u}^\top A\mathbf{v}| \leqslant \|\mathbf{u}^\top A\|_1\|\mathbf{v}\|_\infty.$$

Therefore, we obtain that

$$\|\mathbf{u}^\top A\|_1 \geqslant \frac{\epsilon^4}{3}n^2.$$

Since T consists of either the positive or the negative coordinates of $\mathbf{u}^\top A$, whichever one has larger sum in absolute value, this implies that the T we obtain in step 5b satisfies, with $\mathbf{1}_T$ the characteristic vector,

$$|\mathbf{u}^\top A\mathbf{1}_T| \geqslant \frac{\epsilon^4}{6}n^2.$$

Since $\|\mathbf{u}\|_2 \leqslant \sqrt{n}$, by the Cauchy–Schwarz inequality, this implies that

$$\|A\mathbf{1}_T\|_2^2 \geqslant \frac{(\mathbf{u}^\top A\mathbf{1}_T)^2}{\|\mathbf{u}\|_2^2} \geqslant \frac{\epsilon^8}{36}n^3.$$

Since each row $\mathbf{a}_i$ of A has $\|\mathbf{a}_i\|_2 \leqslant \sqrt{n}$, we also have that

$$\|A\mathbf{1}_T\|_\infty \leqslant \sqrt{n}\|\mathbf{1}_T\|_2 \leqslant n.$$

Therefore,

$$\|A\mathbf{1}_T\|_1 \geqslant \frac{\|A\mathbf{1}_T\|_2^2}{\|A\mathbf{1}_T\|_\infty} \geqslant \frac{\epsilon^8}{36}n^2.$$

This means that for the S that we obtain in step 5b, we have, if $\mathbf{1}_S$ is the characteristic vector,

$$|\mathbf{1}_S^\top A\mathbf{1}_T| \geqslant \frac{\epsilon^8}{72}n^2 \geqslant \frac{\epsilon^8}{100}n^2,$$

which is what we wanted to show. □

We have seen that either output of the algorithm must be correct, so this completes the proof of Theorem 2.2. □

Before proving Theorem 2.1, we need one more technical lemma.

Lemma 2.7. *There exists an $O(n^2)$ time algorithm which takes as input a matrix $A \in \mathbb{R}^{n\times n}$ and subsets $S, T \subseteq [n]$ such that*

$$\sum_{\substack{i\in S\\ k\in T}} a_{i,k} \geqslant \epsilon' n^2,$$

and outputs sets $S', T' \subseteq [n]$ such that

$$\sum_{i\in S', k\in T'} a_{i,k} \geqslant \frac{2}{3}\epsilon' n^2.$$

Furthermore, for any $i \in S'$,

$$\sum_{k\in T'} a_{i,k} \geqslant \frac{\epsilon'}{6} n,$$

and for any $k \in T'$,

$$\sum_{i\in S'} a_{i,k} \geqslant \frac{\epsilon'}{6} n.$$

Proof. Here is the algorithm.

1. To start, set $S' = S$ and $T' = T$.
2. For each $i \in S'$ and each $k \in T'$, store the sum of the corresponding row or column in the submatrix induced by $S' \times T'$.
3. Check whether there is a row or column with sum less than $(\epsilon'/6)n$.
4. If there is, delete it, and update the row or column sums by subtracting the corresponding element from each sum.
5. Go back to step 3 and repeat until no such row or column remains.

We first show that the running time is $O(n^2)$. We can compute each row and column sum in $O(n)$ time, therefore step 2 takes $O(n^2)$ time total. Each time we delete an element from S' or T', we perform $O(n)$ subtractions. The loop runs for at most $2n$ iterations since $|S| + |T| \leqslant 2n$. Thus the algorithm takes $O(n^2)$ time.

We next show that the algorithm is correct. At each step, the sum decreases by at most $(\epsilon'/6)n$, and there are at most $2n$ steps in total. Therefore, after this process, for the S' and T' that we kept, we must still have

$$\sum_{(i,k)\in S\times T} a_{i,k} \geqslant \frac{2}{3}\epsilon' n^2.$$

In particular, this implies that when the algorithm terminates, S' and T' cannot be empty. By the definition of the algorithm, if it terminates, we must have the property that, for any $i \in S'$,

$$\sum_{k\in T'} a_{i,k} \geqslant \frac{\epsilon'}{6} n,$$

and for any $k \in T'$,

$$\sum_{i\in S'} a_{i,k} \geqslant \frac{\epsilon'}{6} n.$$

This completes the proof of the lemma. □

We are now ready to prove our main theorem.

Proof of Theorem 2.1. Let $\epsilon' = \epsilon^8/100$. Here is the algorithm. We iteratively construct a sequence of matrices as follows.

1. Set $A_0 = A$.
2. For each l starting at 0, do the following.
   a. Apply the algorithm from Theorem 2.2 to A_l.
   b. If the algorithm returns that $\|A_l\| \leqslant \epsilon n$, then FINISH.
   c. Otherwise, the algorithm outputs sets $S, T \subseteq [n]$ such that
   $$\left| \sum_{i\in S, k\in T} a_{i,k} \right| \geqslant \epsilon' n^2.$$
   Let $\sigma \in \{-1, 1\}$ be the sign of the above sum.
   d. Use Lemma 2.7, applied to σA_l (and S, T from above), to find $S', T' \subseteq [n]$ such that
   $$\sigma \sum_{i\in S', k\in T'} a_{i,k} \geqslant \frac{2}{3}\epsilon' n^2.$$
   Furthermore, for any $i \in S'$,
   $$\sigma \sum_{k\in T'} a_{i,k} \geqslant \frac{\epsilon'}{6} n,$$
   and for any $k \in T'$,
   $$\sigma \sum_{i\in S'} a_{i,k} \geqslant \frac{\epsilon'}{6} n.$$
   Replace S and T with S' and T'.
   e. Let $S_l = S$, $T_l = T$, $t = \sigma(\epsilon'/3)$, and $A_{l+1} = A_l - tK_{S_l,T_l}$.

Let us first show that we can indeed apply Theorem 2.2 to each A_l. We first show that if $\mathbf{v}$ is a row or column of A_l, then

$$\|\mathbf{v}\|_2^2 \leqslant n.$$

By the assumptions of the theorem, this is true for $l = 0$. Fix l so that it is true for A_l, let $a_{i,k}$ be the entries of A_l, and let i be any row. If $i \notin S$, then the row does not change, so the L^2-norm of the row does not change in A_{l+1}. If $i \in S$, then we have

$$\sum_{k\in T} a_{i,k}^2 - (a_{i,k} - t)^2 = 2t \sum_{k\in T} a_{i,k} - |T|t^2 \geqslant 2t\sigma \frac{\epsilon'}{6} n - t^2 n = t\left(\sigma \frac{\epsilon'}{3} - t\right) n = 0.$$

Since the entries in A_l were $a_{i,k}$, and the entries in A_{l+1} are $a_{i,k} - t$, this implies that the L^2-norm of the corresponding row in A_{l+1} cannot increase, and so for each row it is still at most $\sqrt{n}$. The analogous argument for columns shows that the same holds for each column.

Next, note that each entry of A_0 has absolute value at most 1, and each entry changes by at most $\epsilon'/3$ when going from A_l to A_{l+1}. Therefore, each entry of A_l is at most $1 + l\epsilon'/3$ in absolute value, so we can apply Theorem 2.2 with $C = 1 + l\epsilon'/3$.

Finally, we show that the Frobenius norms of the matrices must decrease:

$$\|A_{l+1}\|_F^2 \leqslant \|A_l\|_F^2 - \frac{\epsilon'^2}{3} n^2.$$

Let $a_{i,k}$ be the entries of A_l again. We have

$$\|A_l\|_F^2 - \|A_{l+1}\|_F^2 = \sum_{\substack{i\in S\\ k\in T}} a_{i,k}^2 - (a_{i,k}-t)^2 = 2t\sum_{\substack{i\in S\\ k\in T}} a_{i,k} - |S||T|t^2$$
$$\geqslant \sigma\frac{4}{3}t\epsilon' n^2 - |S||T|t^2 \geqslant \left(\sigma\frac{4}{3}t\epsilon' - t^2\right)n^2.$$

With our choice of $t=\sigma(\epsilon'/3)$, this implies that

$$\|A_{l+1}\|_F^2 \leqslant \|A_l\|_F^2 - \frac{\epsilon'^2}{3}n^2.$$

Now, we must have $\|A_0\|_F^2 \leqslant n^2$. Since the square of the Frobenius norm decreases by at least

$$\frac{\epsilon'^2}{3}n^2 = \frac{\epsilon^{16}}{30000}n^2$$

at each step, the number of steps is at most $O(1/\epsilon^{16})$. Therefore, after at most $O(1/\epsilon^{16})$, the algorithm must terminate.

As for the running time, the algorithm from Theorem 2.2 (with $C=1+l\epsilon'/3$) takes at most $O((l/\epsilon)^{O(1)}n^2)=\epsilon^{-O(1)}n^2$ time as $l=O(\epsilon^{-16})$. The algorithm from Lemma 2.7 takes $O(n^2)$ time. Finally, as the number of steps is $O(\epsilon^{-16})$, the whole process takes $\epsilon^{-O(1)}n^2$ time. □

3. Approximation algorithm for subgraph counts

We would like to approximate the number of copies of a fixed k-vertex graph H in an n-vertex graph G, up to an additive error of at most ϵn^k. In this section, we prove Theorem 1.4, which claims an algorithm to perform the task in $O(\epsilon^{-O_H(1)}n^2)$ time.

It will be cleaner to work instead with hom (H,G), the number of graph homomorphisms from H to G. This quantity differs from the number of (labelled) copies of H in G by a negligible $O_H(n^{v(H)-1})$ additive error. We use the following multipartite version.

Definition 3.1. Let H be a graph on $[k]$, and let G be a k-partite weighted graph with vertex sets $V_1,\dots,V_k$. We write

$$\hom^*(H,G) = \sum_{(v_1,\dots,v_k)\in V_1\times\cdots\times V_k}\ \prod_{\{i,j\}\in E(H)} G(v_i,v_j), \tag{3.1}$$

where $G(x,y)$ denotes the edge-weight of $\{x,y\}$ in G, as usual.

Note that for graphs H and G, hom$^*(H,G)$ counts the number graph homomorphisms from H to G where every vertex $v_i\in V(H)$ is mapped to the associated vertex part V_i in G.

For every graph G, there is a k-partite G^*, obtained by replicating each vertex of G into k identical copies and two vertices of G^* are adjacent if the original vertices in G they came from are adjacent, such that hom $(H,G)=$ hom$^*(H,G^*)$. Thus Theorem 1.4 follows from its multipartite generalization below.

Theorem 3.2. *There exists a deterministic algorithm that takes as input a graph H on $[k]$, a k-partite graph G with each vertex part having at most n vertices, and $\epsilon>0$, and outputs, in time $\epsilon^{-O_H(1)}n^2$, a quantity that approximates* hom$^*(H,G)$ *up to an additive error of at most ϵn^k.*

Proof. We begin with a description of the algorithm. If H has no edges, then hom$^*(H,G)=|V_1|\cdots|V_k|$. Assume now that H has at least one edge, say $\{1,2\}$ (relabelling if necessary). Denote the vertex parts of G by $V_1, \dots, V_k$. Let G_{12} denote the bipartite graph induced by V_1 and V_2 in G, and $d(G_{12}) = d(V_1, V_2) = e(V_1, V_2)/(|V_1||V_2|)$ to denote the edge density between V_1 and V_2 in G. By Theorem 1.3 and the remark afterwards, we can algorithmically find $S_1, \dots, S_r \subseteq V_1$, $T_1, \dots, T_r \subseteq V_2$, and $c_1, \dots, c_r = O(\epsilon^8)$, with $r = O(\epsilon^{-16})$, such that the weighted bipartite graph G'_{12} on vertex sets V_1 and V_2 defined by

$$G'_{12} = \sum_{i=1}^{r} c_i K_{S_i, T_i} \tag{3.2}$$

satisfies

$$d_\square(G_{12}, G'_{12}) \leqslant \epsilon/2.$$

Let $G^{(i)}$ be obtained from G by deleting the vertices in $(V_1 \setminus S_i) \cup (V_2 \setminus T_i)$. Let H' be H with the edge $\{1, 2\}$ removed. Since H' has one fewer edge than H, we can recursively apply the algorithm to estimate each of $\hom^*(H', G), \hom^*(H', G^{(1)}), \dots, \hom^*(H', G^{(r)})$ up to an additive error of at most $c\epsilon^9$, where c is some absolute constant. Summing up a linear combination of these estimates, we obtain an estimate for

$$\sum_{i=1}^{r} c_i \hom^*(H', G^{(i)}),$$

which we use as our estimate for $\hom^*(H, G)$.

Now we prove the correctness of the algorithm. Let G' be obtained from G by replacing the bipartite graph between V_1 and V_2 by G'_{12}. We claim that

$$\big| \hom^*(H, G) - \hom^*(H, G') \big| \leqslant \frac{\epsilon n^k}{2}. \tag{3.3}$$

Indeed,

$$\hom^*(H, G) - \hom^*(H, G') = \sum_{(v_1, \dots, v_k) \in V_1 \times \cdots \times V_k} f_{v_3, \dots, v_k}(v_1) g_{v_3, \dots, v_k}(v_2) (G(v_1, v_2) - G'(v_1, v_2))$$

for some $f_{v_3, \dots, v_k}(v_1), g_{v_3, \dots, v_k}(v_2) \in \{0, 1\}$ obtained by appropriately grouping the $G(v_i, v_j)$ factors in (3.1).[3] For fixed $(v_3, \dots, v_k) \in V_3 \times \cdots \times V_k$, we have

$$\begin{aligned} &\Bigg| \sum_{(v_1, v_2) \in V_1 \times V_2} f_{v_3, \dots, v_k}(v_1) g_{v_3, \dots, v_k}(v_2)(G(v_1, v_2) - G'(v_1, v_2)) \Bigg| \\ &\qquad \leqslant \max_{U \subset V_1, W \subset V_2} |e_{G_{12}}(U, W) - e_{G'_{12}}(U, W)| \leqslant n^2 d_\square(G_{12}, G'_{12}) \leqslant \epsilon n^2/2. \end{aligned}$$

Then, summing over all $(v_3, \dots, v_k) \in V_3 \times \cdots \times V_k$ and applying the triangle inequality, we obtain (3.3).

From (3.2), we have

$$\hom^*(H, G') = \sum_{i=1}^{r} c_i \hom^*(H', G^{(i)}).$$

Since $c_i = O(\epsilon^8)$ and $r = O(\epsilon^{-16})$, we obtain an estimate of $\hom^*(H, G')$ up to an additive error of at most $\epsilon n^k/2$ as long as each $\hom^*(H', -)$ in the above sum is estimated up to an additive error of $c\epsilon^9 n^k$ for an appropriate positive constant c. Together with (3.3), the estimate is within ϵn^k of $\hom^*(H, G)$, as claimed.

[3]We use the assumption that $0 \leqslant G \leqslant 1$ in this step. In the analogous step in [6], we mistakenly also assumed that $0 \leqslant G' \leqslant 1$, which is not necessarily the case.

Now we analyse the running time. It takes $\epsilon^{-O(1)}n^2$ time (independent of H) to find $S_1, \dots, S_r$, $T_1, \dots, T_r$, and $c_1, \dots, c_r$. Estimating each $\hom^*(H', G)$, $\hom^*(H', G^{(1)}), \dots, \hom^*(H', G^{(r)})$ up to an additive error of at most $c\epsilon^9$ takes $\epsilon^{-O_{H'}(1)}n^2$ time (by induction), and we need to perform $r + 1 = O(\epsilon^{-16})$ such estimates. Thus the total running time is $\epsilon^{-O_H(1)}n^2$. $\square$

References

[1] Alon, N., Duke, R. A., Lefmann, H., Rödl, V. and Yuster, R. (1994) The algorithmic aspects of the regularity lemma. *J. Algorithms* **16** 80–109.
[2] Dellamonica, D., Kalyanasundaram, S., Martin, D., Rödl, V. and Shapira, A. (2012) A deterministic algorithm for the Frieze–Kannan regularity lemma. *SIAM J. Discrete Math.* **26** 15–29.
[3] Dellamonica, D. Jr., Kalyanasundaram, S., Martin, D. M., Rödl, V. and Shapira, A. (2015) An optimal algorithm for finding Frieze–Kannan regular partitions. *Combin. Probab. Comput.* **24** 407–437.
[4] Duke, R. A., Lefmann, H. and Rödl, V. (1995) A fast approximation algorithm for computing the frequencies of subgraphs in a given graph. *SIAM J. Comput.* **24** 598–620.
[5] Fox, J., Lovász, L. M. and Zhao, Y. (2018) Erratum for 'On regularity lemmas and their algorithmic applications'. *Combin. Probab. Comput.* **27** 851–852.
[6] Fox, J., Lovász, L. M. and Zhao, Y. (2017) On regularity lemmas and their algorithmic applications. *Combin. Probab. Comput.* **26** 481–505.
[7] Frieze, A. and Kannan, R. (1996) The regularity lemma and approximation schemes for dense problems. In *37th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, pp. 12–20.
[8] Frieze, A. and Kannan, R. (1999) Quick approximation to matrices and applications. *Combinatorica* **19** 175–220.
[9] Gabber, O. and Galil, Z. (1981) Explicit constructions of linear-sized superconcentrators. *J. Comput. System Sci.* **22** 407–420.
[10] Kohayakawa, Y., Rödl, V. and Thoma, L. (2003) An optimal algorithm for checking regularity. *SIAM J. Comput.* **32** 1210–1235.
[11] Lovász, L. and Szegedy, B. (2007) Szemerédi's lemma for the analyst. *Geom. Funct. Anal.* **17** 252–270.
[12] Lubotzky, A., Phillips, R. and Sarnak, P. (1988) Ramanujan graphs. *Combinatorica* **8** 261–277.
[13] Margulis, G. A. (1973) Explicit constructions of expanders. *Problemy Peredači Informacii* **9** 71–80.
[14] Margulis, G. A. (1988) Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators. *Problemy Peredachi Informatsii* **24** 51–60.
[15] Morgenstern, M. (1994) Existence and explicit constructions of $q + 1$ regular Ramanujan graphs for every prime power q. *J. Combin. Theory Ser. B* **62** 44–62.
[16] Reingold, O., Vadhan, S. and Wigderson, A. (2002) Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Ann. of Math.* (2) **155** 157–187.
[17] Szemerédi, E. (1975) On sets of integers containing no k elements in arithmetic progression. *Acta Arith.* **27** 199–245.
[18] Szemerédi, E. (1978) Regular partitions of graphs. *Problèmes Combinatoires et Théorie des Graphes (Colloq. Internat. CNRS, Univ. Orsay, Orsay, 1976)*, Colloq. Internat. CNRS, Vol. 260, CNRS, Paris, pp. 399–401.

Cite this article: Fox J, Lovász LM and Zhao Y (2019). A fast new algorithm for weak graph regularity. *Combinatorics, Probability and Computing*. **28**, 777–790. https://doi.org/10.1017/S0963548319000075